Copyrighted Material

Contents

Foreword xxi
Preface xxiii
Acknowledgments xxxi
About the Authors xxxiii
Part I: Foundations 1
Chapter 1: The Problem of Delivering Software 3
Introduction 3
Some Common Release Antipatterns 4
Antipattern: Deploying Software Manually 5
Antipattern: Deploying to a Production-like Environment Only after Development Is Complete 7
Antipattern: Manual Configuration Management of Production Environments 9
Can We Do Better? 10
How Do We Achieve Our Goal? 11
Every Change Should Trigger the Feedback Process 13
The Feedback Must Be Received as Soon as Possible 14
The Delivery Team Must Receive Feedback and Then Act on It 15
Does This Process Scale? 16
What Are the Benefits? 17
Empowering Teams 17
Reducing Errors 18
Lowering Stress 20
Deployment Flexibility 21
Practice Makes Perfect 22

Copyrighted Material

Copyrighted Material

The Release Candidate 22
Every Check-in Leads to a Potential Release 23
Principles of Software Delivery 24
Create a Repeatable, Reliable Process for Releasing Software 24
Automate Almost Everything 25
Keep Everything in Version Control 26
If It Hurts, Do It More Frequently, and Bring the Pain Forward 26
Build Quality In 27
Done Means Released 27
Everybody Is Responsible for the Delivery Process 28
Continuous Improvement 28
Summary 29
Chapter 2: Configuration Management 31
Introduction 31
Using Version Control 32
Keep Absolutely Everything in Version Control 33
Check In Regularly to Trunk 35
Use Meaningful Commit Messages 37
Managing Dependencies 38
Managing External Libraries 38
Managing Components 39
Managing Software Configuration 39
Configuration and Flexibility 40
Types of Configuration 41
Managing Application Configuration 43
Managing Configuration across Applications 47
Principles of Managing Application Configuration 47
Managing Your Environments 49
Tools to Manage Environments 53
Managing the Change Process 53
Summary 54
Chapter 3: Continuous Integration 55
Introduction 55
Implementing Continuous Integration 56
What You Need Before You Start 56
A Basic Continuous Integration System 57

Copyrighted Material

Copyrighted Material

Prerequisites for Continuous Integration 59
Check In Regularly 59
Create a Comprehensive Automated Test Suite 60
Keep the Build and Test Process Short 60
Managing Your Development Workspace 62
Using Continuous Integration Software 63
Basic Operation 63
Bells and Whistles 63
Essential Practices 66
Don't Check In on a Broken Build 66
Always Run All Commit Tests Locally before Committing, or Get Your CI Server to Do It for You 66
Wait for Commit Tests to Pass before Moving On 67
Never Go Home on a Broken Build 68
Always Be Prepared to Revert to the Previous Revision 69
Time-Box Fixing before Reverting 70
Don't Comment Out Failing Tests 70
Take Responsibility for All Breakages That Result from Your Changes . 70
Test-Driven Development 71
Suggested Practices 71
Extreme Programming (XP) Development Practices 71
Failing a Build for Architectural Breaches 72
Failing the Build for Slow Tests 73
Failing the Build for Warnings and Code Style Breaches 73
Distributed Teams 75
The Impact on Process 75
Centralized Continuous Integration 76
Technical Issues 76
Alternative Approaches 77
Distributed Version Control Systems 79
Summary 82
Chapter 4: Implementing a Testing Strategy 83
Introduction 83
Types of Tests 84
Business-Facing Tests That Support the Development Process 85
Technology-Facing Tests That Support the Development Process 89
Business-Facing Tests That Critique the Project 89

Copyrighted Material

Copyrighted Material

Technology-Facing Tests That Critique the Project 91
Test Doubles 91
Real-Life Situations and Strategies 92
New Projects 92
Midproject 94
Legacy Systems 95
Integration Testing 96
Process 99
Managing Defect Backlogs 100
Summary 101
Part II: The Deployment Pipeline 103
Chapter 5: Anatomy of the Deployment Pipeline 105
Introduction 105
What Is a Deployment Pipeline? 106
A Basic Deployment Pipeline 111
Deployment Pipeline Practices 113
Only Build Your Binaries Once 113
Deploy the Same Way to Every Environment 115
Smoke-Test Your Deployments 117
Deploy into a Copy of Production 117
Each Change Should Propagate through the Pipeline Instantly 118
If Any Part of the Pipeline Fails, Stop the Line 119
The Commit Stage 120
Commit Stage Best Practices 121
The Automated Acceptance Test Gate 122
Automated Acceptance Test Best Practices 124
Subsequent Test Stages 126
Manual Testing 128
Nonfunctional Testing 128
Preparing to Release 128
Automating Deployment and Release 129
Backing Out Changes 131
Building on Success 132
Implementing a Deployment Pipeline 133
Modeling Your Value Stream and Creating a Walking Skeleton 133
Automating the Build and Deployment Process 134

Copyrighted Material

Copyrighted Material

Automating the Unit Tests and Code Analysis 135
Automating Acceptance Tests 136
Evolving Your Pipeline 136
Metrics 137
Summary 140
Chapter 6: Build and Deployment Scripting 143
Introduction 143
An Overview of Build Tools 144
Make 146
Ant 147
NAnt and MSBuild 148
Maven 149
Rake 150
Buildr 151
Psake 151
Principles and Practices of Build and Deployment Scripting 152
Create a Script for Each Stage in Your Deployment Pipeline 152
Use an Appropriate Technology to Deploy Your Application 152
Use the Same Scripts to Deploy to Every Environment 153
Use Your Operating System's Packaging Tools 154
Ensure the Deployment Process Is Idempotent 155
Evolve Your Deployment System Incrementally 157
Project Structure for Applications That Target the JVM 157
Project Layout 157
Deployment Scripting 160
Deploying and Testing Layers 162
Testing Your Environment's Configuration 163
Tips and Tricks 164
Always Use Relative Paths 164
Eliminate Manual Steps 165
Build In Traceability from Binaries to Version Control 165
Don't Check Binaries into Version Control as Part of Your Build 166
Test Targets Should Not Fail the Build 166
Constrain Your Application with Integrated Smoke Tests 167
.NET Tips and Tricks 167
Summary 168

Copyrighted Material

Copyrighted Material

Chapter 7: The Commit Stage 169
Introduction 169
Commit Stage Principles and Practices 170
Provide Fast, Useful Feedback 171
What Should Break the Commit Stage? 172
Tend the Commit Stage Carefully 172
Give Developers Ownership 173
Use a Build Master for Very Large Teams 174
The Results of the Commit Stage 174
The Artifact Repository 175
Commit Test Suite Principles and Practices 177
Avoid the User Interface 178
Use Dependency Injection 179
Avoid the Database 179
Avoid Asynchrony in Unit Tests 180
Using Test Doubles 180
Minimizing State in Tests 183
Faking Time 184
Brute Force 185
Summary 185
Chapter 8: Automated Acceptance Testing 187
Introduction 187
Why Is Automated Acceptance Testing Essential? 188
How to Create Maintainable Acceptance Test Suites 190
Testing against the GUI 192
Creating Acceptance Tests 193
The Role of Analysts and Testers 193
Analysis on Iterative Projects 193
Acceptance Criteria as Executable Specifications 195
The Application Driver Layer 198
How to Express Your Acceptance Criteria 200
The Window Driver Pattern: Decoupling the Tests from the GUI 201
Implementing Acceptance Tests 204
State in Acceptance Tests 204
Process Boundaries, Encapsulation, and Testing 206
Managing Asynchrony and Timeouts 207
Using Test Doubles 210

Copyrighted Material

Copyrighted Material

The Acceptance Test Stage 213
Keeping Acceptance Tests Green 214
Deployment Tests 217
Acceptance Test Performance 218
Refactor Common Tasks 219
Share Expensive Resources 219
Parallel Testing 220
Using Compute Grids 220
Summary 222

Chapter 9: Testing Nonfunctional Requirements 225
Introduction 225
Managing Nonfunctional Requirements 226
Analyzing Nonfunctional Requirements 227
Programming for Capacity 228
Measuring Capacity 231
How Should Success and Failure Be Defined for Capacity Tests? 232
The Capacity-Testing Environment 234
Automating Capacity Testing 238
Capacity Testing via the User Interface 240
Recording Interactions against a Service or Public API 241
Using Recorded Interaction Templates 241
Using Capacity Test Stubs to Develop Tests 244
Adding Capacity Tests to the Deployment Pipeline 244
Additional Benefits of a Capacity Test System 247
Summary 248

Chapter 10: Deploying and Releasing Applications 249
Introduction 249
Creating a Release Strategy 250
The Release Plan 251
Releasing Products 252
Deploying and Promoting Your Application 253
The First Deployment 253
Modeling Your Release Process and Promoting Builds 254
Promoting Configuration 257
Orchestration 258
Deployments to Staging Environments 258

Copyrighted Material

Copyrighted Material

Rolling Back Deployments and Zero-Downtime Releases 259
Rolling Back by Redeploying the Previous Good Version 260
Zero-Downtime Releases 260
Blue-Green Deployments 261
Canary Releasing 263
Emergency Fixes 265
Continuous Deployment 266
Continuously Releasing User-Installed Software 267
Tips and Tricks 270
The People Who Do the Deployment Should Be Involved in Creating the Deployment Process 270
Log Deployment Activities 271
Don't Delete the Old Files, Move Them 271
Deployment Is the Whole Team's Responsibility 271
Server Applications Should Not Have GUIs 271
Have a Warm-Up Period for a New Deployment 272
Fail Fast 273
Don't Make Changes Directly on the Production Environment 273
Summary 273
Part III: The Delivery Ecosystem 275
Chapter 11: Managing Infrastructure and Environments 277
Introduction 277
Understanding the Needs of the Operations Team 279
Documentation and Auditing 280
Alerts for Abnormal Events 281
IT Service Continuity Planning 282
Use the Technology the Operations Team Is Familiar With 282
Modeling and Managing Infrastructure 283
Controlling Access to Your Infrastructure 285
Making Changes to Infrastructure 287
Managing Server Provisioning and Configuration 288
Provisioning Servers 288
Ongoing Management of Servers 290
Managing the Configuration of Middleware 295
Managing Configuration 296
Research the Product 298
Examine How Your Middleware Handles State 298

Copyrighted Material

Copyrighted Material

Look for a Configuration API 299
Use a Better Technology 299
Managing Infrastructure Services 300
Multihomed Systems 301
Virtualization 303
Managing Virtual Environments 305
Virtual Environments and the Deployment Pipeline 308
Highly Parallel Testing with Virtual Environments 310
Cloud Computing 312
Infrastructure in the Cloud 313
Platforms in the Cloud 314
One Size Doesn't Have to Fit All 315
Criticisms of Cloud Computing 316
Monitoring Infrastructure and Applications 317
Collecting Data 318
Logging 320
Creating Dashboards 321
Behavior-Driven Monitoring 323
Summary 323
Chapter 12: Managing Data 325
Introduction 325
Database Scripting 326
Initializing Databases 327
Incremental Change 327
Versioning Your Database 328
Managing Orchestrated Changes 329
Rolling Back Databases and Zero-Downtime Releases 331
Rolling Back without Losing Data 331
Decoupling Application Deployment from Database Migration 333
Managing Test Data 334
Faking the Database for Unit Tests 335
Managing the Coupling between Tests and Data 336
Test Isolation 337
Setup and Tear Down 337
Coherent Test Scenarios 337
Data Management and the Deployment Pipeline 338
Data in Commit Stage Tests 338

Copyrighted Material

Copyrighted Material

Data in Acceptance Tests 339
Data in Capacity Tests 341
Data in Other Test Stages 342
Summary 343
Chapter 13: Managing Components and Dependencies 345
Introduction 345
Keeping Your Application Releasable 346
Hide New Functionality Until It Is Finished 347
Make All Changes Incrementally 349
Branch by Abstraction 349
Dependencies 351
Dependency Hell 352
Managing Libraries 354
Components 356
How to Divide a Codebase into Components 356
Pipelining Components 360
The Integration Pipeline 361
Managing Dependency Graphs 363
Building Dependency Graphs 363
Pipelining Dependency Graphs 365
When Should We Trigger Builds? 369
Cautious Optimism 370
Circular Dependencies 372
Managing Binaries 373
How an Artifact Repository Should Work 373
How Your Deployment Pipeline Should Interact with the Artifact Repository 374
Managing Dependencies with Maven 375
Maven Dependency Refactorings 377
Summary 379
Chapter 14: Advanced Version Control 381
Introduction 381
A Brief History of Revision Control 382
CVS 382
Subversion 383
Commercial Version Control Systems 385
Switch Off Pessimistic Locking 386

Copyrighted Material

Copyrighted Material

Branching and Merging 388
Merging 389
Branches, Streams, and Continuous Integration 390
Distributed Version Control Systems 393
What Is a Distributed Version Control System? 393
A Brief History of Distributed Version Control Systems 395
Distributed Version Control Systems in Corporate Environments 396
Using Distributed Version Control Systems 397
Stream-Based Version Control Systems 399
What Is a Stream-Based Version Control System? 399
Development Models with Streams 400
Static and Dynamic Views 403
Continuous Integration with Stream-Based Version Control Systems ... 403
Develop on Mainline 405
Making Complex Changes without Branching 406
Branch for Release 408
Branch by Feature 410
Branch by Team 412
Summary 415
Chapter 15: Managing Continuous Delivery 417
Introduction 417
A Maturity Model for Configuration and Release Management 419
How to Use the Maturity Model 419
Project Lifecycle 421
Identification 422
Inception 423
Initiation 424
Develop and Release 425
Operation 428
A Risk Management Process 429
Risk Management 101 429
Risk Management Timeline 430
How to Do a Risk-Management Exercise 431
Common Delivery Problems—Their Symptoms and Causes 432
Infrequent or Buggy Deployments 433
Poor Application Quality 434
Poorly Managed Continuous Integration Process 435

Copyrighted Material

Copyrighted Material

Poor Configuration Management 436
Compliance and Auditing 436
Automation over Documentation 437
Enforcing Traceability 438
Working in Silos 439
Change Management 440
Summary 442
Bibliography 443
Index 445

Copyrighted Material